by Don Fudge

Shape Sequence Animation

Last month I began a discussion of animation by describing how to effect scrolling on the screen. That launches us into a look at shape sequence animation.

With vector shapes you can use whatever shape table numbers you want, in whatever order you want, and any number of shapes can occur in a sequence. For example, suppose you wanted to make a stick-figure man "walk." You might have a sequence of 4, 7, 6, 8, 1, 9, 3, 5, 2, 7 for shape numbers of your sequence shapes. That ten sequence-shape numbers, with repeating allowed. Shape numbers are referring to Applesoft shape numbers which get numbers because of their shape table index. You needn't use shapes' numbers from assembly, but in Basic it's the only convenient way to DRAW or XDRAW. It might be more convenient to have your shape table numbers be the same as the shape sequence numbers. One thing that makes this not particularly important is the fact that you'll often use specific shapes more than once in a sequence.

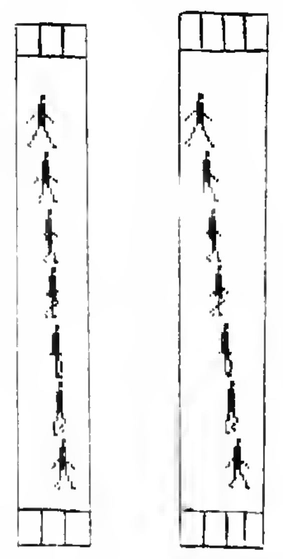


Figure 1. Printout of block shape sequences.

Walking

Think of walking. There are a couple of times in a walking sequence when, viewed from the side, one specific shape could represent more than one specific aspect of the sequence. There's no sin in using the number 4 shape twice, for example. So an algorithm to have a stick figure walk will be constructed like so:

1) Erase, by XDRAW, the shape at old coordinates (OX,OY).
2) Draw, by XDRAW, the shape at new coordinates (X,Y).
3) Dump the new coordinates into the old coordinates (OX = X:OY = Y).
4) Calculate the new coordinates using a step value, X = X + STEP. If the figure is moving vertically as well as horizontally, such as walking upstairs, also do Y = Y + STEP.
5) Go back to 1.

Remember that if you're doing page flipping things will be more complex and you'll be drawing on one screen while displaying the next. The fundamentals of this method were covered in my April column. Page flipping is a way to stop showing the drawing *process* and begin showing the drawing *results* only. The effect of this is to smooth things out and make the animation not look flickery.

Which Screen, Which Shape, Draw and Erase Chart

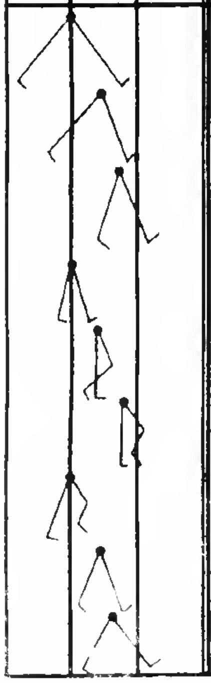
screen on which to erase/draw	screen displayed	shape # erased	shape # drawn	HL of shape erased	HL of shape drawn
1	2	1	3	0	0
2	1	2	4	0	0
1	2	3	5	0	0
2	1	4	6	0	0
1	2	5	7	0	0
2	1	6	1	0	1
1	2	7	2	0	1
2	1	1	3	1	1
1	2	2	4	1	1
2	1	3	5	1	1
1	2	4	6	1	1
2	1	5	7	1	1
1	2	6	1	1	2
2	1	7	2	1	2
1	2	1	3	2	2
↓	↓	↓	↓	↓	↓
2	1	5	7	34	34
1	2	6	1	34	0
2	1	7	2	34	0
1	2	1	3	0	0

Table. Which Screen, Which Shape, Draw and Erase chart

Address correspondence to Don Fudge at Avant-Garde Creations, PO Box 30160, Eugene, OR 97403.

3-byte wide block shape

MANA

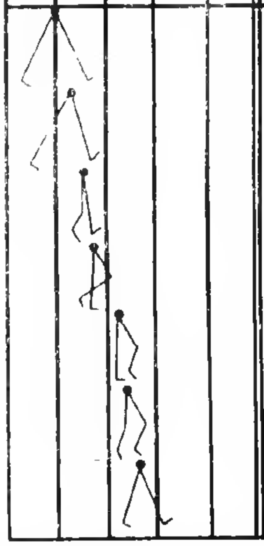


1st byte	2nd byte	3rd byte	sequence #	shape # from MAN	centered on this hor coord	VT VB	move HL and HR up by 1?	hor coords of block-shape boundaries	MANC seq #
			1	1	35	19 40	yes	28-49	-
			2	6	38	19,40	no	28-49	1
			3	2	40	19,40	no	28 49	2+7
			4	7	42	19 40	yes	35 56	3
			5	3	45	19 40	no	35 56	4
			6	8	47	19 40	no	35 56	5
			7	4	49	19,40	yes	42 63	6
			8	9	52	19 40	no	42 63	-
			9	10	54	19 40	no	42 63	-

Figure 2a. MANA. Incrementing the horizontal byte column (X) coordinate by the step value three times per sequence

5-byte wide block shape

MANC



1st byte	2nd byte	3rd byte	4th byte	5th byte	seq #	shape #	shape centered on hor coord	VT VB	move HL and HR up by 2?	hor coords of block shape boundaries
					1	6	35	0 21	yes	28 63
					2	2	37	0 21	no	28 63
					3	7	39	0 21	no	28 63
					4	3	41	0 21	no	28 63
					5	8	43	0 21	no	28 63
					6	4	45	0 21	no	28 63
					7	2	47	0 21	no	28 63

Figure 2b. MANC. Adding the step value to the horizontal byte column (X) coordinate when the sequence is finished

This is a good place to discuss block shape sequences. With block shapes, it's not just a matter of drawing proper shape sequences in the proper places at the proper times and incrementing by a constant step value for the next coordinate. It's true that you can place vector shapes anywhere on the screen at any time, with illegal positions at X<0,X>279,Y<0 and Y>191 (so use these for parameter checking). But block shapes cannot be handled likewise.

With block shapes you must stay within Y=0 and Y=191 and also X byte column (horizontal offset) 0 and 39. And you can't move less than 1 byte horizontally if you have only one shape, unless you want to use relatively slow *shift animation*. See *Hi-Res Secrets* for details on that. So, you'll almost always be using what's known as *pre-shifted shapes*, in sequences of seven.

Pre-shifted shape sequences are block shape sequences that allow less-than-7-dot (1 visible byte) moves horizontally. For similar graphics objects, such as seven identical flying saucers, pre-shifted shapes are a simple matter of running an automatic sequence creator (Listing 1) on the first flying saucer and saving the resultant seven-shape sequence as a table. Take a look at Figures 1, 2a and 2b.

In Figure 1 we see a step 1 (per move), seven-shape block shape sequence that is 3 bytes wide, and a step 2, seven-shape block shape sequence that is 4 bytes wide. Consider the left and right boundaries of these shape blocks to be the actual block shape boundaries. Notice how throughout the running of a shape sequence, neither the X coordinate nor the Y coordinate changes one iota. It is only when the sequence is finished that we add the step value to the horizontal byte-column coordinate. This is illustrated in Figure 2b. In Figure 2a, however, the X coordinate is incremented three times per sequence. In both diagrams, HR means horizontal right coordinate, HL means horizontal left coordinate, VT means vertical top coordinate, and VB means vertical bottom coordinate:

```
     VT
  •••••••
HL•••••••HR
  •••••••
  •••••••
     VB
```

Again, block shapes have only 40 possible X coordinates per screen, not 280 like vector shapes, because block shapes use byte-column coordinates, not regular X coordinates, in the horizontal direction.

Block Shape Sequences

In Figure 1, shapes 1–10 were extracted from a vector shape table (MAN) to create the nine shapes in MANA's block shape sequence table, which was updated three times per sequence in a very non-standard way. But from MANA was created MANC, a standard seven-shape ever-incrementing sequence of block shapes (Figure 2a). All it took was

Listing 1. Sequence Creator.

```
0  ONERR  GOTO 63990
1  PRINT  CHR$ (4);"BLOADTEST H (CALL2186)": GOSUB 2500: GOTO 600
2  NONE : INPUT "SHAPE TABLE NAME: ";STN$: IF  LEN (STN$) = 0 THEN 600
4 O$ =  CHR$ (4): PRINT O$"BLOAD",STN$
5  HOME : VTAB 21: INPUT "SHAPE #: ";SHN: POKE 7,SHN
15  POKE  - 16304,0: POKE  - 16297,0
18 VS = I:BS = 0
20  INPUT "VTOP:";VT: INPUT "VBOT:";VB: INPUT "HRIGHT:",HR: INPUT "HLEFT:"
     ,HL
30  POKE 252,VT: POKE 253,VB: POKE 254,HR: POKE 255,HL
42  CALL 2116
43  HOME : VTAB 21: INPUT "DO YOU WANT ANOTHER SHAPE? (Y/N):",QH$: IF  LEN
     (QH$) = 0 THEN 43
44  IF  ASC (QH$) <  > 89 THEN  HOME : VTAB 21: GOSUB 63000: GOTO 600
45  GOTO 5
47  POKE  - 16303,0: POKE  - 16298,0: HOME : VTAB I: PRINT "USE THE PADDLE
     S TO MOVE THE DOT TO THE  UPPER LEFT RECTANGLE POINT. HIT PDL 0   BUT
     TON. TNEN MOVE THE OOT TO TNE LOWER  RIGHT RECTANGLE POINT. HIT PDL 1
      BUTTON.": GOSUB 63000
48  POKE 232,248: POKE 233,8: SCALE= 1: ROT= 64
49  POKE  - I6304,0: POKE  - 16297,0
50  HONE :P1 =  PDL (1): IF P1 > 150 TNEN 50
55 P0 =  PDL (0): XDRAW 1 AT P0,P1:X% = P0:Y% = P1
60 P1 =  POL (1): IF P1 > I50 THEN 60
65  FOR QH = 1 TO 200: NEXT : HOME : VTAB 21: PRINT "X: "P0: PRINT "Y: "P1

70 P0 =  PDL (0): XDRAW 1 AT X%,Y%: XDRAW 1 AT P0,P1:X% = P0:Y% = P1
80 B0 =  PEEK ( - 16287): IF B0 > 127 AND FL = 0 THEN FL = 1: GOTO 100
85 B1 =  PEEK ( - 16286): IF B1 > 127 ANO SG = 0 THEN SG = 1: GOTO 110
90  GOTO 60
100 VT = P1:HL =  1NT (P0 / 7): PRINT  CHR$ (7): IF SG = 1 THEN 120
105  GOTO 60
110 VB = P1:HR =  1NT (P0 / 7): PRINT  CHR$ (7): IF FL = 1 THEN 120
115  GOTO 60
120  HOME : VTAB 21: PRINT "HOR.--FROM:"HL" TO "HR"---HIDTH:"(HR - HL)
125  XDRAW 1 AT P0,P1
130  PRINT "VER.--FROM:"VT" TD "VB"---HEIGHT:"(VB - VT): VTAB 23: PRINT "J
     OT THIS DOWN! (HIT ANY KEY TO CONT.)": GOSUB 63010
150  POKE 252,VT: POKE 253,VBOT: POKE 254,HRIGHT: POKE 255,HLEFT
155  HCOLOR= 3
160  HPLOT 7 * NRIGNT + 7,VT TO 7 + HR1GHT + 7,VB TO 7 + HLEFT,VB TO 7 + H
     LEFT,UT TO 7 + HR1GHT + 7,UT
170  IF ZQ = 1 THEN  RETURN
```

Listing continued

Circle 302 on Reader Service card.

LOCK-IT-UP

DISKETTE COPY-PROTECTION
AND DUPLICATION SYSTEMS
FOR THE APPLE COMPUTER

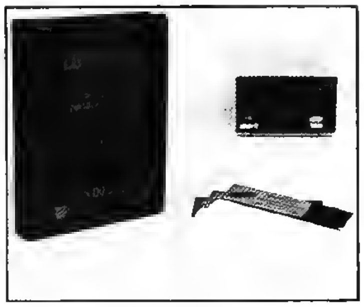
The Lock-It-Up systems are sophisticated, menu driven copy-protection and duplication utilities for the Apple II Computer. They feature several levels of protection which make standard diskettes uncopyable by even the most sophisticated nibble copy programs currently available.

- All sectors on the diskette can still be used.
- Data files can be loaded and/or saved to either the protected diskette or an unprotected diskette.
- Memory will be cleared and the disk will reboot if the reset key is pushed.
- The copying systems support up to 14 disk drives.
- Complete data-verification is optional during copy.
- Sequential serial numbers are assigned to each diskette produced by the system
- Master diskettes created with the system contain an I.D. stamp that you select. The I.D. stamp must be correctly specified before any diskettes can be duplicated. This prevents other Lock-It-Up owners from copying your diskettes.
- Extensive support is provided should you have any problems or special needs.
- Our system is supported by numerous disk copying services should you need a large quantity of diskettes duplicated.

Either system is available for $225, which includes three diskettes, an informative manual, and a non-exclusive license to copy as many diskettes as needed.

ORDER NOW! Call collect for COD, Mastercard or Visa orders

DEALER INQUIRIES INVITED



DOS VERSION:

- Any standard DOS 3.3 diskette can be protected.
- DOS command names can be changed and/or deleted.
- Autorun can be used to prevent the listing of a program or the use of any basic commands outside of a program.
- A faster DOS can be used in order to decrease disk access time by up to 50%!

REQUIRES: 48K Apple II or II+ with Applesoft in ROM or language system and at least two disk drives.

PASCAL VERSION:

- Any standard Apple Pascal 1.1 diskette can be protected
- Files may be transferred to a standard Pascal diskette, but they will not run unless they are on the protected diskette.
- Easily added to any program by use of a Regular Unit.
- Compatible with Apple Fortran.

REQUIRES: Apple Pascal and at least two disk drives.

DOUBLE - GOLD
SOFTWARE

13126 ANZA DRIVE
SARATOGA, CA 95070
(408) 257-2247

Listing continued.

```
180  PRINT : INPUT "IS THE RECTANGLE OONE O.K.? (Y/N):";AN$: IF  LEN (AN$)
      = 0 THEN 180
185  IF  ASC (AN$) = 78 THEN SG = 0: NCOLOR= 0:FL = 0:ZQ = 1: GOSU0 160:ZQ
      = 0: HCOLOR= 3: GOTO 50
191  GOTO 600
204  HOME : VTAB 21
205  PRINT "SNAPE # "ST
200  POKE 7,ST: POKE  - 16304,0: POKE  - 16297,0
210  IF QZ = 0 THEN QZ = 1:ZQ = 1: HCOLOR= 0: GOSUB 160: HCOLOR= H
215 NN = NN + 1
220  CALL 2048
225  IF NN >  = NS THEN 300
240  FOR QO = 1 TO SS: CALL 2186: NEXT
245 ST = ST + 1
250  GOTO 204
300 O$ =  CHR$ (4)
301  VTAB 21
302  INPUT "FILE NAME: ";N$: IF  LEN (N$) = 0 THEN 302
303  IHPUT "010 YOU GET IT RIGHT? (Y/N):";Z$: IF  LEN (Z$) = 0 THEN 302
304  IF  ASC (Z$) <  > 89 TNEN 302
307  TEXT : VTA0 1: HOHE : 6OSU0 5040
308 LL = 256 * LS
309 PL = LS
310  PRINT D$"BSAVE";N$;",A2304,L";LL
312  PRINT "LAST SHAPE ANO ALL THE SHAPES THAT CAHE BEFORE IT TOOK UP "LL"
      BYTES.": PRINT "LAST SHAPE: ";LS: PRINT "(HIT ANY KEY TO CONTINUE):"
     : GOSUB 63010
400  GOTO 600
402  HOME : VTA0 21: INPUT "STEP SIZE: ";SS
403  PRINT : INPUT "# OF SHAPES IN SEQUENCE: ";NS: PRINT : INPUT "# OF 1ST
      0LOCK-SHAPE IN SEQUENCE TO 0E  SAVED: ";ST
404  PRINT : INPUT "READY TO 0EGIN AUTOMATIC SCAN & SAVE    PROCESS FOR TH
     IS SEQUENCE? (Y/N):";QH$: IF  LEN (QH$) = 0 THEN 600
405  IF  ASC (QH$) <  > 89 THEN 600
410  GOTO 204
600  POKE  - 16303,0: POKE  - 16298,0: HOME : VTA0 1: INVERSE : HTAB 18: PRINT
     "MENU:": NORMAL
601 SG = 0:FL = 0:ZQ = 0:QZ = 0:NN = 0
602  SCALE= S: HCOLOR= H: ROT= R
603  PRINT "(HIT ESC TO QUIT)": PRINT
605  PRINT "(0)ABORT SCREEN---START OVER": PRINT
610  PRINT "(1)LOAO 0LOCK SHAPE TABLE": PRINT
640  PRINT "(2)6IVE HOR. STEP SIZE FOR 0LOCK-SHAPE     SEQUENCE & SAVE ENT
     IRE SEQUENCE": PRINT
650  PRINT "(3)DEFINE BLOCK SHAPE NITH PADDLES": PRINT
660  PRINT "(4)VIEN SCREEN": PRINT
690  FLASH : PRINT "(CHOOSE 0-4):";: NORMAl  : GET A$: PRINT  CHR$ (13)
692  IF  ASC (A$) = 27 THEN  TEXT : HOME : END
700  IF  LEN (A$) = 0 THEN 690
710  IF  VAL (A$) < 0 OR  VAL (A$) > 4 THEN 690
719  IF A$ = "0" TNEN 912
720  ON  VAL (A$) 6OTO 2,402,47,920,600
721  GOTO 600
912  INPUT "SURE YOU HANT TO A0ORT SCREEN? (Y/N):";QH$: IF  LEN (QH$) = 0 THEN
     912
913  IF  ASC (QH$) <  > 89 THEN 600
914  HGR : GOTO 600
920  POKE  - 16304,0: POKE  - 16297,0: VTA0 21: 6OSU0 63000: GOTO 600
2500  POKE 2296,1: POKE 2297,0: POKE 2298,4: POKE 2299,0: POKE 2300,4: POKE
     2301,0
2510  POKE  - 16301,0
2511  POKE  - 16303,0: POKE  - 16298,0: INVERSE : PRINT "IF YOU ENTERED TH
     IS PROGRAM WITH SOME-  THING ON THE NI-RES SCREEN YOU HANTEO TOSAVE,
     HIT THE SPACE 0AR NOH---              OTNERHISE HIT ANY KEY EXCEPT THE SP
     ACE  BAR.": NORMAL
2512 PK =  PEEK ( - 16384): IF PK > 127 TNEN  POKE  - 16360,0: GOTO 2514
2513  GOTO 2512
2514  IF PK = 160 THEN 2520
2515  HGR
2520  RETURN
5040  HOME : VTA0 21: INPUT "# OF LAST SHAPE IN 0LOCK-SHAPE TABLE: ";LS: IF
     LS < 1 OR LS > 23 THEN 5040
5050  RETURN
63000  PRINT "                                          HIT ANY KEY TO CONT
     INUE:           "
63010 PK =  PEEK ( - 16384): IF PK > 127 THEN  POKE  - 16360,0: RETURN
63020  GOTO 63010
63990  PRINT  CHR$ (7): POKE 216,0
63991 PP =  PEEK (222): IF PP = 254 THEN  RESUME
63994  POKE  - 16303,0: POKE  - 16290,0
63995  PRINT "YUUR ERROR IS COUE #:"PP: 6OSUB 63000: CALL 54915: GOTO 600
```

loading various vector and block shapes into SCANA (see the April column) and saving them at pre-calculated coordinates (saving them as various shape table numbers).

Looking at Figure 2a again, notice that the block shapes are 5 bytes wide, but could just as easily have been 4 bytes wide. (The extra "blank" byte was used for experimental purposes.) Now, look at the first and seventh shapes. Where would we put the next (eighth) shape if we were to continue the sequence, and what would it look like?

Well, first notice that each shape is being moved 2 dots to the right of the previous one. Then observe that we'll be looking for a shape like sequence number 1 to continue the "movement." Also note that X=49 will be the horizontal coordinate of the center of the next shape, so the first shape in the sequence will end up centered exactly on the line again, just as it is in its diagram position. Since the figure in the block shape sequence will move over exactly 2 bytes and the step value, in dots, between each of the figures in the shape sequence is 2, then that means the step value is equal to the required horizontal byte-coordinate increment we'll be using just before starting the sequence over.

What this means is that during the display of the seven shapes shown, all block shape coordinates stay exactly the same. It's only just before the sequence restart that the horizontal byte coordinate gets increased by 2. So what's happening, in effect, is that most of the movements of the block shape figures take place within the boundaries of the block shape, and not by coordinate manipulation. Incidentally, all shape numbers given in MANA and MANC are taken from MAN, a vector shape table for a man walking. The actual shape numbers you'll refer to as you build and use a block shape table such as MANC are shapes 1–7, equivalent to sequence numbers 1–7.

Two-Page Flipping

When you use pre-shifted shapes of the block shape sequence variety and then go for unflickering smoothness by use of two-page flipping animation, the level of complexity goes up by several orders of magnitude. Check out the table and you'll see that things can get awkward pretty quickly. You draw on one screen and display the other. One screen will get the sequence 1, 3, 5, 7, 2, 4, 6, 1, 3, etc., while the other screen is getting, alternately, 2, 4, 6, 1, 3, 5, 7, etc.

The strangest part is when <single asterisk> HL <horizontal left byte coordinate> is 0 while erasing shape 6, 1 while drawing shape 1, back to 0 for erasing shape 7 on the opposite screen (double asterisks) and up to 1 again for drawing shape 2. It's important to keep good charts of what's happening when coding such animation routines.

```
0  ONERR  GOTO 63990
1  POKE 8,0: REM  8 MUST BE 0E0 FOR THIS PROG. TO WORK!!!!!!!!!!!!!!
5  HIMEM: 36864
10 D$ =  CHR$ (4)
25  TEXT : INPUT "SHAPE TABLE NAME: ";N$: IF  LEN (N$) = 0 THEN 25
27  PRINT : INPUT "YOU WANT YOUR SHAPE TO TRAVEL:            (1) ----  RIGHT
    WARDS                         (2) <---- LEFTWARDS                      (1-2
    ):";Q: IF Q < 1 OR Q  > 2 THEN 27
28  IF Q = 1 THEN  PRINT  CHR$ (4);"BLOADTEST F (CALL36934)"
29  IF Q = 2 THEN  PRINT  CHR$ (4);"BLOADTEST G (CALL36934)"
30  PRINT : PRINT "INSERT YOUR SHAPE TABLE DISK NOW:": GET A$: PRINT  CHR$
    (13): CALL 1002: PRINT D$"BLOAD";N$: PRINT "ADDRESS: " PEEK (43634) +
     PEEK (43635) * 256: PRINT "LENGTH: " PEEK (43616) +  PEEK (43617) *
    256
31  INPUT "WIDTH:";WD: INPUT "HEIGHT:";HT: INPUT "STEP SIZE:";SS: INPUT "R
    IGHT BOUNDARY OF LEFT SIDE OF SHAPE:";RB: IF Q = 1 THEN  POKE 235,WD -
    SS: POKE 29,(RB - SS) + WD: POKE 30,RB - SS
32  INPUT "# OF 1ST SHAPE IN SEQUENCE: ";SH: POKE 239,WD: POKE 238,RB: POKE
    237,HT: POKE 236,SS: IF Q = 2 THEN  POKE 25,39 - WD: POKE 235,SS + 39

33  IF Q = 1 THEN  POKE 36955,SH + 1: POKE 36987,SH: POKE 36994,SH + 1: POKE
    37029,SH + 5: POKE 37046,SH: POKE 37080,SH - 1: POKE 37084,SH + 6: POKE
    37133,SH + 1
34  IF Q = 2 THEN  POKE 37092,SH + 7: POKE 37096,SH: POKE 37152,SH + 5: POKE
    36955,SH + 5: POKE 36907,SH + 6: POKE 36994,SH + 5: POKE 37029,SH + 1
    : POKE 37846,SH + 6
35  TEXT : INPUT "DELAY LOOP HI BYTE (1-255):";A: IF A < 1 OR A > 255 THEN
    35
36  POKE 9,A
37  PRINT : INPUT "DELAY LOOP LO BYTE (1-255):";B: IF B < 1 OR B       THEN
    37
38  POKE 31,B
40  CALL 36934
42  HOME
43  TEXT : PRINT "$EF:" PEEK (239): PRINT "$EE:" PEEK (238): PRINT "$ED:" PEEK
    (237): PRINT "$EC:" PEEK (236): PRINT "$EB:" PEEK (235): PRINT "$1D:"
     PEEK (29): PRINT "$1E:" PEEK (30): PRINT : PRINT
45  TEXT : INPUT "DO YOU WANT TO SEE IT SOME MORE? (Y/N):";Q$: IF  LEN (Q$
    ) = 0 THEN 45
46  IF  ASC (Q$) <  > 89 THEN  END
50  GOTO 31
63990  POKE 216,0
63991  ONERR  GOTO 63990
63992 PK =  PEEK (222): IF PK = 254 THEN  RESUME
63995  GOTO 0
```

Listing 2. ASMINPUT.

Now, if you key in TEST F(CALL 36934) and TEST G (CALL36934) in Listings 4 and 5, and then MANC and ASMINPUT in Listings 3 and 2, you'll have a two-page flipping block shape sequence using routine for moving left or right using seven-shape block sequences. Here are some BSAVE addresses and lengths for various files in these listings:

Listing 3	MANC,A$900,L1646 (step 2, 21 high, 4 wide, 7 shapes)
Listing 4	TEST F (CALL36934), A 36864, L324
Listing 5	TEST G (CALL36934), A 36864, L342
Listing 6	TEST H (CALL2186), A 2048, L224

When keying in MANC, ignore the data, such as from $970 to $9FF, that's omitted and key in only data given.

I recommend POKE 103,1: POKE 104,96: POKE 24576,0 in your Hello (boot) program before running any of the programs in this article.

Let's "make the man walk" by animating the seven shapes in MANC with the TEST F (CALL36934) and TEST G (CALL36934) animation routines. (These routines effect right and left

```
+900.96F

0900- 00 00 00 00 00 00 E0 00
0908- 30 08 00 00 00 10 [illegible] 00
0910- 00 00 08 08 00 00 00 00
0918- 10 00 00 00 04 10 00 00
0920- 00 04 20 00 00 00 02 20
0928- 00 00 00 02 40 0E 00 00
0930- 03 40 00 00 00 13 44 00
0938- 00 00 0B 48 00 00 00 07
0940- 50 00 00 00 83 60 00 00
0948- 00 03 40 00 00 00 03 40
0950- 00 00 00 03 40 00 00 00
0958- 01 00 00 00 00 03 00 00
0960- 00 00 03 00 00 00 00 00
0968- 00 00 00 00 00 00 BB 0B
+A00.A6F

0A00- 00 00 00 00 00 00 00 00
0A08- 00 00 00 00 00 61 00 00
0A10- 00 00 20 40 00 00 00 21
0A18- 00 00 00 00 12 00 00 00
0A20- 00 12 00 00 00 00 0A 00
0A28- 00 00 00 0A 00 00 00 00
0A30- 4E 20 00 00 00 2E 40 00
0A38- 00 00 1E 40 00 00 00 0F
0A40- 00 00 00 00 0E 00 00 00
0A48- 00 0E 0A 00 00 00 0E 00
0A50- 00 00 00 0E 00 00 00 00
0A58- 04 00 00 00 00 0C 00 00
0A60- 00 00 0C 00 00 00 00 00
0A68- 00 00 00 00 00 00 00 00
+B00.E6F

0B00- 00 00 00 00 00 00 00 01
0808- 48 00 00 00 00 44 00 00
```

```
0B10- 00 00 48 00 00 00 00 50
0B18- 60 00 00 00 30 00 00 00
0B20- 00 30 00 00 00 00 30 00
0B28- 00 00 00 30 00 00 00 00
0B30- 38 00 00 00 01 3A 00 00
0B38- 00 00 7C 00 00 00 00 38
0B40- 00 00 00 00 38 00 00 00
0B48- 00 38 00 00 00 00 38 00
0B50- 00 00 00 38 00 00 00 00
0B58- 10 00 00 00 00 30 00 00
0B60- 00 00 30 00 00 00 00 00
0B68- 00 00 00 00 00 00 00 00
+C00.C6F
0C00- 00 00 00 00 00 00 00 00
0C08- 00 00 00 00 01 50 00 00
0C18- 00 00 00 00 60 00 00 00
0C20- 01 40 00 00 00 02 40 00
0C28- 00 00 01 40 00 00 00 01
0C30- 68 00 00 00 05 70 00 00
0C38- 00 03 60 00 00 00 01 60
0C40- 00 00 00 01 60 00 00 00
0C48- 01 60 00 00 00 01 60 00
0C50- 00 00 01 60 00 00 00 00
0C58- 40 00 00 00 01 40 00 00
0C60- 00 01 40 00 00 00 00 00
0C68- 00 00 00 00 00 00 00 00
+D00.06F

0D00- 00 08 00 00 00 00 00 0E
0D08- 00 00 00 00 06 00 00 06
0D10- 00 0A 00 00 00 00 0A 00
0D10- 00 00 00 0A 00 00 00 00
0D20- 0A 00 00 00 00 06 00 00
0D28- 00 00 06 00 00 00 00 0F
0D30- 40 00 00 00 07 00 00 00
0D38- 00 07 00 00 00 00 07 00
0D40- 00 00 00 07 00 00 00 00
```

```
0D48- 07 00 00 00 00 07 00 00
0D50- 00 00 07 00 00 00 00 02
0D58- 00 00 00 00 06 00 00 00
0D60- 00 06 00 00 00 00 00 00
0D60- 00 00 00 00 00 00 00 00
+E00.E6F

0E00- 00 00 00 00 00 00 00 06
0E08- 00 00 00 00 12 00 00 00
0E10- 00 0A 00 00 00 00 12 00
0E18- 00 00 00 22 00 00 00 00
0E20- 24 00 00 00 00 14 00 00
0E28- 00 00 14 00 00 00 00 1C
0E30- 00 00 00 00 1C 00 00 00
0E38- 00 1C 00 00 00 00 1C 00
0E40- 00 00 00 1C 00 00 00 00
0E48- 1C 00 00 00 00 1C 00 00
0E50- 00 00 1C 00 00 00 00 00
0E58- 00 00 00 00 18 00 00 00
0E60- 00 18 00 00 00 00 00 00
0E68- 00 00 00 00 00 00 00 00
+F00.F6F

0F00- 00 00 00 00 00 00 00 00
0F08- 00 00 00 06 08 00 00 00
0F10- 02 04 00 00 00 02 08 00
0F18- 00 00 01 10 00 00 00 01
0F20- 10 00 00 00 00 50 00 00
0F28- 00 00 50 00 00 00 04 72
0F30- 00 00 00 02 74 00 00 00
0F38- 01 74 00 00 00 00 78 00
0F40- 00 00 00 70 00 00 00 00
0F40- 70 00 00 00 00 70 00 00
0F50- 00 00 70 00 00 00 00 20
0F58- 00 00 00 00 60 00 00 00
0F60- 80 60 00 00 00 00 00 00
0F68- 00 00 00 00 00 00 00 00
```

Listing 3. MANC.

```
*9000.9144

9000- A0 09 A6 07 CA E0 00 F0
9008- 04 C8 4C 04 90 90 85 FB
9010- A9 00 85 FA A5 FD 85 06
9018- A2 00 A0 00 20 11 F4 A4
9020- FE A2 00 A1 FA 51 26 91
9028- 26 00 18 E6 FA D0 02 E6
9030- F8 C0 FF F0 04 C4 FF 00
9038- EA C6 06 A5 06 C9 FF F0
9040- 04 C5 FC 00 D3 60 20 E2
9048- F3 A9 00 8D 52 C0 20 D8
9050- F3 A9 40 85 E6 A9 00 8D
9058- 54 C0 A9 02 85 07 A9 00
9060- 85 FF A5 EF 85 FE A9 00
9068- 85 FC A5 ED 85 FD 20 00
9070- 90 A9 00 8D 55 C0 A9 20
9078- 85 E6 A9 01 85 07 20 00
9080- 90 A9 02 85 07 A9 20 85
9088- E6 A9 00 8D 55 C0 20 A0
9090- 90 A9 40 85 E6 A9 00 8D
9098- 54 C0 20 A0 90 4C 85 90
90A0- C6 07 A5 07 C9 06 D0 2D
90A8- 20 00 90 A4 09 A6 1F CA
90B0- D0 FD 88 D0 FB A9 01 85
90B8- 07 A4 EC E6 FE E6 FF 88
90C0- D0 F9 10 A5 FF C5 EE 90
90C8- 08 A5 EF 85 FE A9 00 85
90D0- FF 20 00 90 60 A5 07 C9
90D8- 00 D0 48 A9 07 85 07 A4
90E0- EC C6 FE C6 FF 88 D0 F9
90E8- A5 FE C5 EB D0 08 A5 1E
90F0- 85 FF A5 1D 85 FE 20 00
90F8- 90 A4 09 A6 1F CA D0 FD
9100- 88 D0 F8 A4 EC E6 FE E6
9108- FF 88 D0 F9 A9 02 85 07
9110- 10 A5 FF C5 EE 90 08 A5
9110- EF 85 FE A9 00 85 FF 20
9120- 00 90 60 20 00 90 A4 09
9128- A6 1F CA D0 FD 88 D0 F0
9138- E6 07 E6 07 20 00 90 10
9138- AD 00 C0 C9 88 B0 01 60
9140- 60 20 60 00 00
```

Listing 4. TEST F (CALL36934).

```
9000- A0 09 A6 07 CA E0 00 F0
9008- 04 C8 4C 04 90 98 05 FB
9010- A9 00 05 FA A5 FD 05 06
9010- A2 00 A0 00 20 11 F4 A4
9020- FE A2 00 A1 FA 51 26 91
9028- 26 00 10 E6 FA D0 02 E6
9030- F8 C0 FF F0 04 C4 FF 00
9038- EA C6 06 A5 06 C9 FF F0
9040- 04 C5 FC 00 D3 60 20 E2
9048- F3 A9 00 0D 52 C0 20 D8
9050- F3 A9 40 85 E6 A9 00 8D
9058- 54 C0 A9 06 85 07 A5 19
9060- 85 FF A9 27 85 FE A9 00
9068- 85 FC A5 ED 85 FD 20 00
9070- 90 A9 00 8D 55 C0 A9 20
9078- 85 E6 A9 07 85 07 20 00
9080- 90 A9 06 85 07 A9 20 85
9088- E6 A9 00 0D 55 C0 20 A0
9090- 90 A9 40 85 E6 A9 00 8D
9098- 54 C0 20 A0 90 4C 85 90
90A0- E6 07 A5 07 C9 02 D0 39
90A8- 20 00 90 A4 09 A6 1F CA
90B0- D0 FD 88 D0 F0 A9 07 85
90B8- 07 A4 EC C6 FE C6 FF 88
90C0- D0 F9 A5 FE C5 EF B0 15
90C8- 10 A5 FE 65 EC 85 1D A5
90D0- FF 65 EC 85 1E A9 27 85
90D8- FE A5 19 85 FF 20 00 90
90E0- 60 A5 07 C9 08 D0 4E A9
90E8- 01 85 07 A4 EC E6 FE E6
90F0- FF 88 D0 F9 A5 FE C5 EB
90F8- D0 0F 38 A5 1D E5 EC 85
9100- 08 A5 1E 85 FF A5 1D 85
9108- FE 20 00 90 A4 09 A6 1F
9110- CA D0 FD 88 D0 F0 A4 EC
9110- C6 FE C6 FF 88 D0 F9 A9
9120- 06 85 07 A5 FE C5 00 D0
9128- 88 A9 27 85 FE A5 19 85
9130- FF 20 00 90 60 20 00 90
9138- A4 09 A6 1F CA D0 FD 88
9140- D0 F0 C6 07 C6 07 20 00
9140- 90 10 AD 00 C0 C9 80 B0
9150- 01 60 68 20 60 00 00 00
```

Listing 5. TEST G (CALL36934).

```
*000.BE2

0800- A0 09 A6 07 CA E0 00 F0
0800- 04 C8 4C 04 08 98 85 FB
0810- A9 00 85 FA A5 FD 85 06
0810- A2 00 A0 88 20 11 F4 A4
0820- FE A2 00 01 26 01 FA 00
0828- 10 E6 FA D0 02 E6 F0 C0
0830- FF F0 04 C4 FF B0 EC C6
0838- 06 A5 06 C9 FF F0 04 C5
0840- FC 00 D5 60 A0 09 A6 07
0840- CA E0 00 F0 04 C8 4C 48
0850- 00 90 85 FB A9 00 85 FA
0858- A5 FD 85 06 A2 00 A0 00
0860- 20 11 F4 A4 FE A2 00 A1
0868- FA 51 26 91 26 88 10 E6
0870- FA D0 02 E6 F8 C0 FF F0
0878- 04 C4 FF 88 EA C6 06 A5
0880- 06 C9 FF F0 04 C5 FC 88
0888- D3 60 A5 FD 85 06 A2 00
0890- A0 00 20 11 F4 10 A4 FF
0898- A9 00 85 00 85 CF B1 26
08A0- 2A 91 26 B0 02 90 02 E6
08A8- 88 C9 80 B0 02 90 02 E6
0880- CF A5 00 D0 09 01 26 29
0888- 7F 91 26 4C C4 08 01 26
08C0- 09 00 91 26 C4 FE F0 09
08C8- C8 10 A5 CF C9 01 4C 98
08D0- 08 C6 06 A5 06 C9 FF F0
08D8- 04 C5 FC 08 01 60 00 00
```

Listing 6. TEST H (CALL2186).

movement.) The Basic *driver* program we'll use (to be RUN now) is ASMINPUT. Give the shape table name of MANC. (I'm assuming you've saved the necessary files.) Say RIGHTWARDS for direction of travel. Specify a width of 4, a height of 21, a step size of

Circle 237 on Reader Service card.

How to Choose the Best Modem For Your Apple®: Now SSM Offers You THE SOURCE℠

Features	Hayes Micromodem II™	Novation AppleCat II™	SSM Apple ModemCard™
110/300 baud operation	Yes	Yes	Yes
Supports Apple //e 80-Column Card	No	No	Yes
Half/Full Duplex	Yes	Yes	Yes
Auto-Dial/Auto-Answer	Yes	Yes	Yes
Fits completely inside Apple	No	No	Yes
Touch-Tone® Dialing	No	Yes	Yes
"Single-Modem-Chip" Reliability	No	No	Yes
Audio Monitor	No	No	Yes
Self Testing	Yes	Yes	Yes
Warranty period	2 yr	1 yr	2 yr
Includes SOURCE Offer	No	No	Yes
Suggested Retail Price	$379	$389	$325

Trademarks: Micromodem Hayes Microcomputer Products, Inc. Applecat II Novation. ModemCard SSM Microcomputer Products Inc. The Source: Source Telecomputing Corporation, a subsidiary of the Reader's Digest Association. Dow Jones News/Retrieval Dow Jones & Company, Inc. Apple is a registered trademark of Apple Corporation.

Count the Yes-es. SSM's Apple ModemCard: the most advanced features for the least money. No external equipment required. All other modems are now obsolete.

And with SSM's Transend software, your Apple can send electronic mail or talk to other computers or information services such as the Source and Dow Jones News/Retrieval®. (The ModemCard is also compatible with software for the Micromodem II).

The SSM ModemCard. The only choice for choosy Apple owners. Available from your local computer dealer. Satisfaction is guaranteed. Or your money back.

SSM Microcomputer Products Inc
2190 Paragon Drive, San Jose,
CA 95131 (408) 946-7400,
Telex 171171 SSM SNJ

Circle 103 on Reader Service card

The House-Ware Genies

- ★ The Mail Genie
- ★ The Pantry Genie
- ★ The Recipe Genie
- ★ The Calendar Genie
- ★ The Insurance Genie

At last, the Turn-Key Programs that allow Dad to influence Mom, that an Apple™ Computer is a modern day necessity.

Introductory Price $29.95

Developed By
SOUTHERN CENTER
FOR RESEARCH & INNOVATION
P.O. Box 1713
Hattiesburg, MS 39403

Dealer Inquiries Invited

Circle 102 on Reader Service card

LOOK at these prices!

AMDEK COLOR 1 13"	$289
HAYES SMARTMODEM 1200	499
HAYES MICROMODEM II	265
SATURN 128K CARD	425
SATURN ACCELERATOR II	499
MICROSOFT MULTIPLAN	180
VIDEX 80 COL CARD VIDEOTERM	227
VIDEX ULTRATERM	339
ORANGE MICRO GRAPPLER +	125
INTERACTIVE PHOTO CARD	135
EASTSIDE WILDCARD COPY DEVICE	108
CALIF COMP SERIAL ASYNCH CARD	129
R H SUPER FAN II	70
R H SUPER FAN II ZENER RAY	99

[illegible] Money Order Cashiers Check Bank Wire [illegible] Checks Allow 2 weeks clearance COD's [illegible] Certified Checks only NY Res Add 8 [illegible]% ST
Add $3.00 Shipping Charge on all orders
Please include your phone number

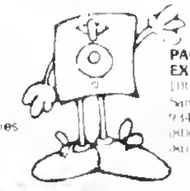
MICRO MYSTIQUE
73- [illegible] 183rd St, Flushing NY 1136[illegible]
212 454-619[illegible]
Call or Write for Additional Specials

[illegible] Reader Service card

Scotch Diskettes

[illegible] Scotch® diskettes to keep your valuable [illegible] safe. Dependable Scotch diskettes are tested and guaranteed error free. The low abrasivity saves your read/write heads. [illegible] compatible with most diskette drives.

(800)235-4137



Dealer Inquiries Invited

PACIFIC EXCHANGES
100 Foothill Blvd
San Luis Obispo CA
93401. In Ca. call
(800) 592-5935 or
(805) 543-1037

2, a right boundary of 34, and a first shape in sequence of 1. Then give a delay loop high byte of 70 and a delay loop low byte of 255.

This results in a realistic walking speed and no flicker problems (due to the two-page flipping animation in these routines), but the two-step movements are somewhat noticeable. A good way to improve this would be to have 14 shapes in the sequence and do one-step movements, still incrementing the horizontal byte coordinate by 2 at the end of the sequence, but using only one half the delay time. Notice that when HR (horizontal right coordinate) gets to 34 the sequence restarts. That's the 34 you input above. Hit any key and do it again, but use a delay loop high byte of 35 so things move twice as fast. (Ignore the $EE:34 and other data given on the screen; this just indicates how the variables are doing once the action stops.)

Now he's marching along. You can see that the step movements within block shape parameters are no longer noticeable. Now try a step value of 1 and a delay loop high byte of 70 again. Notice how block shape sequences meant for a step value of 2 do weird things with a value of 1. But also notice that the intra-block step movements, being only 1 dot each, are much less obvious with a step value of 1. These latter two experiments should support the idea that 14 one-step shapes with a horizontal byte coordinate increment of 2 will yield the smoothest results. (Incidentally, when moving to the left, choose LEFTWARDS, but for the other inputs choose the same as you did for RIGHTWARDS.)

Sequence Creation

Now key in TEST II (CALL2186) and SEQUENCE CREATOR, Listings 6 and 1. (Don't forget about the POKE 104,96, etc., as advised previously.) Then RUN SEQUENCE CREATOR, hitting return upon entry into the program. Choose (1) LOAD IN BLOCK SHAPE TABLE and give MANC as the shape table name. Then specify shape number 1, VTOP of 10, VBOT of 31, HRIGHT of 5, HLEFT of 1, and no, you don't need any more shapes (when asked).

Now, in the menu, choose (3) DEFINE BLOCK SHAPE WITH PADDLES, read the instructions, and move the paddles to find out which one makes the dot cursor move horizontally. We'll call this paddle your X paddle and the other your Y paddle. Move the dot cursor just outside the upper left corner of the imaginary rectangular block around the man shape, and hit the X paddle button. Now move to the lower right corner and hit the Y paddle button—*but not until you've moved at least 7 dots to the right of that position, to make room for intra-block step movements.* (Use a 14-dot offset if your step value is 2 and a 21-dot offset if your step value is 3, and so on.) Seven times the step value to the right (lower) of your block shape is where you'll hit the Y button.

When asked if the rectangle (which defines the block shape parameters) is okay, answer yes or no. No gets you another chance. Now choose (2) GIVE HORIZONTAL STEP SIZE FOR BLOCK-SHAPE SEQUENCE & SAVE ENTIRE SEQUENCE, and specify 1 for step size, 7 for number of shapes in sequence, 1 for number of first block shape in sequence to be saved, and Y (yes) for "Are you ready for this sequence?" Keep your eyes peeled, and you'll see all seven shapes made by shifting (after which each in turn will be scanned and the resultant data saved in memory). When asked for file name use TEST and give 7 as the number of the last shape in the shape table.

Once the sequence is saved, use it when you RUN ASMINPUT to check the latter out. Step size must be 1, unless you used something greater than that in your sequence creation. Unless you've made a mistake the man will float very smoothly.

If all this sounds like it's right up your alley, drop me a line for more information on routines and utilities for graphics, sounds and more.

Next time I'll dissect the fastest color-fill algorithm around to show how it works. You'll get a chance to save it, a program to use it with and a palette of colors for posterity. See you then! ■